Expressing Computer Science Concepts Through Kodu Game Lab

Kathryn T. Stolee  Teale Fristoe

March 10, 2011
Learning to program is hard
Learning to program is hard
Teaching programming is hard, too
Educational Programming Languages:

- Useful to introduce novices to programming
- Commonly used in a classroom setting
- Used to create video games, simulations, animations, art ...
- Focus is on ease-of-use and attractiveness
- Many examples: Alice, Greenfoot, Scratch ...
About Kodu

- A video game for creating 3d video games
- Designed to compete with modern console games
- Available on Xbox or PC
- Uses an Xbox controller as the interface for playing and programming
- 110,000+ installs in 150+ countries
How Kodu is Different

- Integrates common gaming concepts (e.g., scoring, camera positioning, termination conditions)
- Programming model does not resemble syntax and abstraction level of mainstream languages
How Kodu is Different

- Integrates common gaming concepts (e.g., scoring, camera positioning, termination conditions)
- Programming model does not resemble syntax and abstraction level of mainstream languages

Little is known about how skills learned in Kodu may transfer to more traditional languages.
Interacting with Kodu

**Play Mode**
- Play the game
- Test or simulate programming logic

**Edit Mode**
- Modify terrain and objects
- Program characters
Interacting with Kodu

**Play Mode**
- Play the game
- Test or simulate programming logic

**Edit Mode**
- Modify terrain and objects
- Program characters
Interacting with Kodu

Play Mode
- Play the game
- Test or simulate programming logic

Edit Mode
- Modify terrain and objects
- Program characters
Programming: Kodu Language

- Language is a high-level, visual, and event-driven
- Statements take the form of 'When – Do’ clauses
Programming: Kodu Language

- Language is a high-level, visual, and event-driven
- Statements take the form of 'When – Do’ clauses

Rule → Condition Action
Programming: Kodu Language

- Language is a high-level, visual, and event-driven
- Statements take the form of 'When – Do' clauses

![Rule → Condition Action]

**Rule** → **Condition** Action
Programming: Kodu Language

- Language is a high-level, visual, and event-driven
- Statements take the form of 'When – Do’ clauses

Rule → Condition Action
Programming: Kodu Language

- Language is a high-level, visual, and event-driven
- Statements take the form of 'When – Do' clauses

Rule $\rightarrow$ Condition Action
Programming involves visual composition of tiles
Programming: Rule Prioritization

- Rules are ordered and organized into pages
- Conflicting rules resolve action using order
Characters can have up to 12 pages of programming
Switching pages changes character behavior (e.g., a power-up)
Can traditional computer science concepts (e.g., boolean logic, objects, variables, iteration, control flow) be expressed in Kodu?
Research Questions

RQ1: Which computer science concepts can be expressed through the Kodu Language?

RQ2: How often does each computer science concept appear in the programs created by the Kodu community?
Research Questions

RQ1: Which computer science concepts can be expressed through the Kodu Language?
Kodu Language Analysis

RQ2: How often does each computer science concept appear in the programs created by the Kodu community?
Research Questions

**RQ1:** Which computer science concepts can be expressed through the Kodu Language?

**Kodu Language Analysis**

**RQ2:** How often does each computer science concept appear in the programs created by the Kodu community?

**Analysis of ~350 Kodu Programs**
CS Concepts in Kodu

Obviously Supported:
- Objects: Encapsulation, Creation, Deletion
- Control Structures: if – then, iteration
- Variables: Global, Local, Random
- Boolean Logic: Negation

Subtly Supported:
- Objects: Cloning (class system)
- Boolean Logic: Conjunction, Disjunction
- Control Flow: Cycles, Fan-in, Fan-out

Yet to be Investigated:
- Function calls
- …
CS Concepts in Kodu

Obviously Supported:
- Objects: Encapsulation, Creation, Deletion
- Control Structures: if – then, iteration
- Variables: Global, Local, Random
- **Boolean Logic: Negation**

Subtly Supported:
- Objects: Cloning (class system)
- **Boolean Logic: Conjunction, Disjunction**
- **Control Flow: Cycles, Fan-in, Fan-out**

Yet to be Investigated:
- Function calls
- …
Boolean Logic: Negation

not A
Boolean Logic: Negation

\[ \text{not } A \]

1. when A do B
2. when \text{not } A do C
Boolean Logic: Negation

not A

1 when A do B
2 when not A do C
Boolean Logic: Conjunction

- Condition
- Action

1 when A do B
2 when C do D
3 when always do E
Boolean Logic: Conjunction

- **Condition**
  \[ A \land C \Rightarrow D \]
- **Action**

1. when \( A \) do \( B \)
2. when \( C \) do \( D \)
3. when always do \( E \)
Boolean Logic: Conjunction

- **Condition**
  \[ A \land C \implies D \]

- **Action**
  \[ A \implies B \land E \]

1. when \( A \) do \( B \)
2. when \( C \) do \( D \)
3. when *always* do \( E \)
Boolean Logic: Conjunction

- **Condition**
  \[ A \land C \Rightarrow D \]

- **Action**
  \[ A \Rightarrow B \land E \]

1. when A do B
2. when C do D
3. when always do E
Boolean Logic: Disjunction

\[ A \lor C \Rightarrow B \]
**Boolean Logic: Disjunction**

\[ A \lor C \Rightarrow B \]

1. when \( A \) do \( B \)
2. when \( C \) do \( B \)
Boolean Logic: Disjunction

\[ A \lor C \implies B \]

1. when \( A \) do \( B \)
2. when \( C \) do \( B \)
Expressions of Control Flow

- Pages represent character state in terms of behavior
- Switching pages can create non-linear program flow
Expressions of Control Flow

- Pages represent character state in terms of behavior
- Switching pages can create non-linear program flow
Expressions of Control Flow

- Pages represent character state in terms of behavior
- Switching pages can create non-linear program flow

Illustration showing two control flow scenarios:
1. **Switching pages can create non-linear program flow**
2. **Expressions of control flow**

Diagram with two states, s and 1, connected by arrows labeled "bump coin" and "bump tree."
Control Flow Patterns

Cycles (iteration)
Control Flow Patterns

Cycles (iteration)

Fan-in (reuse)
Control Flow Patterns

Cycles (iteration)

Fan-in (reuse)

Fan-out (conditional flow)
**Research Questions**

**RQ1:** Which computer science concepts can be expressed through the Kodu Language?

**Kodu Language Analysis**

**RQ2:** How often does each computer science concept appear in the programs created by the Kodu community?

**Analysis of ~350 Kodu Programs**
Profile for Kodu Program

346 programs from the XBox community (13 months of data)

<table>
<thead>
<tr>
<th>Property</th>
<th>Average</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rules</td>
<td>109</td>
<td>54</td>
</tr>
<tr>
<td>Total Tiles</td>
<td>497</td>
<td>231</td>
</tr>
<tr>
<td>Programmed Characters</td>
<td>18</td>
<td>12</td>
</tr>
</tbody>
</table>
## Boolean Logic in the Community

<table>
<thead>
<tr>
<th>CS Concept</th>
<th># Games</th>
<th>% Games</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical <em>Not</em></td>
<td>61</td>
<td>17.6%</td>
</tr>
<tr>
<td>If-Then-Else Statements</td>
<td>29</td>
<td>8.4%</td>
</tr>
<tr>
<td>Logical <em>And</em> Condition(^1)</td>
<td>17</td>
<td>20.9%</td>
</tr>
<tr>
<td>Logical <em>And</em> Action(^1)</td>
<td>13</td>
<td>16.0%</td>
</tr>
<tr>
<td>Logical <em>Or</em></td>
<td>208</td>
<td>59.9%</td>
</tr>
</tbody>
</table>

\(^1\) The indentation feature needed for the logical *and* was introduced on March 19, 2010. These values consider only the 81 (23.4%) games published after that date.
## Boolean Logic in the Community

<table>
<thead>
<tr>
<th>CS Concept</th>
<th># Games</th>
<th>% Games</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical <em>Not</em></td>
<td>61</td>
<td>17.6%</td>
</tr>
<tr>
<td>If-Then-Else Statements</td>
<td>29</td>
<td>8.4%</td>
</tr>
<tr>
<td>Logical <em>And</em> Condition</td>
<td>17</td>
<td>20.9%</td>
</tr>
<tr>
<td>Logical <em>And</em> Action</td>
<td>13</td>
<td>16.0%</td>
</tr>
<tr>
<td>Logical <em>Or</em></td>
<td>208</td>
<td>59.9%</td>
</tr>
</tbody>
</table>

1The indentation feature needed for the logical *and* was introduced on March 19, 2010. These values consider only the 81 (23.4%) games published after that date.
# Control Flow in the Community

<table>
<thead>
<tr>
<th>CS Concept</th>
<th># Games</th>
<th>% Games</th>
</tr>
</thead>
<tbody>
<tr>
<td>2+ States</td>
<td>202</td>
<td>58.4%</td>
</tr>
<tr>
<td>Cycles</td>
<td>166</td>
<td>47.9%</td>
</tr>
<tr>
<td>Fan-in &gt; 1</td>
<td>174</td>
<td>50.3%</td>
</tr>
<tr>
<td>Fan-out &gt; 1</td>
<td>123</td>
<td>35.5%</td>
</tr>
</tbody>
</table>
## Control Flow in the Community

<table>
<thead>
<tr>
<th>CS Concept</th>
<th># Games</th>
<th>% Games</th>
</tr>
</thead>
<tbody>
<tr>
<td>2+ States</td>
<td>202</td>
<td>58.4%</td>
</tr>
<tr>
<td>Cycles</td>
<td>166</td>
<td>47.9%</td>
</tr>
<tr>
<td>Fan-in &gt; 1</td>
<td>174</td>
<td>50.3%</td>
</tr>
<tr>
<td>Fan-out &gt; 1</td>
<td>123</td>
<td>35.5%</td>
</tr>
</tbody>
</table>
## Control Flow in the Community

<table>
<thead>
<tr>
<th>CS Concept</th>
<th># Games</th>
<th>% Games</th>
</tr>
</thead>
<tbody>
<tr>
<td>2+ States</td>
<td>202</td>
<td>58.4%</td>
</tr>
<tr>
<td>Cycles</td>
<td>166</td>
<td>47.9%</td>
</tr>
<tr>
<td>Fan-in &gt; 1</td>
<td>174</td>
<td>50.3%</td>
</tr>
<tr>
<td>Fan-out &gt; 1</td>
<td>123</td>
<td>35.5%</td>
</tr>
</tbody>
</table>
Threats to Validity

**Internal** Relationship between *expressing* concepts in code and *learning* those concepts is unverified.

**External** Programs were self-selected for uploading to Xbox community and may not be representative of the broader population.
What We Learned:
- Many fundamental computer science concepts *can* be expressed in Kodu
- Users make extensive use of the language and complex language constructs

Questions Left Unanswered:
- Are Kodu users internalizing the computer science concepts that appear in their games? (Requires evaluation)
- Can Kodu be used to teach computer science?
Expressing Computer Science Concepts Through Kodu Game Lab

Kathryn T. Stolee  Teale Fristoe

kstolee@cse.unl.edu

teale@soe.ucsc.edu

March 10, 2011